WHAT IS CLAIMED IS:

1. A software tool that determines at least one data address from one or more instruction instances, and that identifies one or more memory reference objects, associated with the data address, as hindering execution of code that includes the instruction instances, wherein the instructions instances correspond to the code execution hindrance.

2. The software tool of claim 1 wherein the memory reference objects include one or more of physical memory reference objects and logical memory reference objects.

3. The software tool of claim 2 wherein the physical memory reference objects include one or more of cache, cache lines, cache levels, cache sub-blocks, memory controllers, addressable memory, and memory-management page translation units.

4. The software tool of claim 3, wherein the addressable memory includes one or more of virtually addressable memory and physically addressable memory.

5. The software tool of claim 2 wherein the logical memory reference objects include one or more of source-level data objects, memory segments, heap variables, variable instances, and stack variables.

6. The software tool of claim 5 wherein the source-level data objects include one or more of functions, statically linked objects, data structures, data types, data type definitions, operands, and expressions.

7. The software tool of claim 6 wherein the statically linked objects include one or more of global variables and static variables.

8. The software tool of claim 1 wherein the software tool includes one or more of a compiler, an interpreter, an optimization tool, and a virtual machine.

9. The software tool of claim 1 wherein the code includes one or more of machine code, byte code, and interpreted code.

10. The software tool of claim 1 that also aggregates addresses based on the memory reference objects.

11. The software tool of claim 10 wherein the software tool utilizes at least a portion of the data addresses to aggregate the addresses.

12. The software tool of claim 10 that also provides the aggregated addresses and an indication of the code execution hindrance corresponding to the aggregated addresses for one or more of storage and display.

13. The software tool of claim 1 wherein the data address includes a virtual address or a physical address.

14. The software tool of claim 1 wherein the code execution hindrance corresponds to one or more sampled runtime events.

15. The software tool of claim 14 wherein the sampled runtime events include one or more of cache misses, cache references, data translation buffer misses, data translation buffer references, and counter condition events.

16. A method for profiling code, the method comprising:
identifying an instruction instance that corresponds to a runtime event;
determining a data address from the instruction instance; and
determining a memory reference object from the determined address.

17. The method of claim 16 wherein the runtime event is a sampled runtime event.

18. The method of claim 16 wherein identifying the instruction instance comprises backtracking from a second instruction instance to the instruction instance.

19. The method of claim 16 wherein determining the address from the instruction instance comprises decoding the instruction instance.

20. The method of claim 19 further comprising:

decoding the instruction instance if a register that hosts the instruction instance is determined as valid.

21. The method of claim 20 wherein determining if the register is valid comprises:

applying reverse register transformation with respect to the runtime event; and

determining whether the register is valid based on the applied reverse register transformation.

22. The method of claim 16 wherein the memory reference object includes a physical memory reference object or a logical memory reference object.

23. The method of claim 22 wherein the physical memory reference object includes cache, a cache line, a cache sub-block, a cache level, a memory controller, or a memory-management page translation unit.

24. The method of claim 22 wherein the logical memory reference object includes a source-level data object, a memory segment, a heap variable, or a stack variable.

25. The method of claim 24 wherein the source-level data object includes a data type, a data type definition, a statically linked object, an operand, a data structure, or an expression.

26. The method of claim 25 wherein the statically linked object includes a global variable or a static variable.

27. The method of claim 16 wherein the instruction instances include memory accessing instructions.

28. The method of claim 16 wherein the data address includes a virtual address or a physical address.

29. The method of claim 16 further comprising aggregating a plurality of addresses that include the determined address, based on the memory reference object.

30. The method of claim 29 wherein the aggregating of the plurality of addresses utilizes at least a portion of the addresses.

31. The method of claim 29 further comprising providing the aggregated plurality of addresses for one or more of display, storage, and manipulation.

32. The method of claim 16 embodied as a computer program product encoded on one or more machine-readable media.

33. A method of profiling code, the method comprising:

associating data addresses with memory reference objects, wherein the data addresses have been determined from instruction instances corresponding to code execution hindrance; and

aggregating the data addresses based on their associated memory reference objects.

34. The method of claim 33 wherein the instruction instances include memory accessing instructions.

35. The method of claim 33 wherein the code execution hindrance corresponds to one or more runtime events.

36. The method of claim 35 wherein the runtime events are sampled runtime events.

37. The method of claim 35 wherein the runtime events include one or more of counter condition events, cache misses, cache references, data translation buffer references, and data translation buffer misses.

38. The method of claim 33 wherein the data addresses include one or more of virtual addresses and physical addresses.

39. The method of claim 33 wherein said aggregating utilizes at least a portion of the data addresses.

40. The method of claim 33 embodied as a computer program product encoded on one or more machine-readable media.

41. A method of profiling code comprising:

identifying an instruction instance corresponding to a runtime event;

determining whether the instruction instance is valid;

decoding the instruction instance to extract at least a portion of a data address if the instruction instance is valid;

determining a memory reference object with the extracted portion of the address; and

aggregating the data address with other addresses based at least in part on the memory reference object.

42. The method of claim 41 further comprising associating the extracted portion of the data address with the memory reference object.

43. The method of claim 41 wherein the addresses include one or more of physical addresses and virtual addresses.

44. The method of claim 41 wherein the runtime event is a sampled runtime event.

45. The method of claim 41 further comprising:

applying reverse register transformation with respect to the runtime event to determine if the instruction instance is valid.

46. The method of claim 41 embodied as a computer program product encoded on one or more machine-readable media.

47. A computer program product for profiling code, encoded on one or more machine-readable media, the computer program product, which when executed, performs operations comprising:

identifying a valid instruction instance that corresponds to a runtime event;

determining a data address from the identified valid instruction instance;

determining a memory reference object with the determined data address; and

aggregating a set of addresses, which include the determined data address, based at least in part on the memory reference object.

48. The computer program product of claim 47 wherein the operations further comprise associating the determined data address with the memory reference object.

49. The computer program product of claim 47 wherein the operations further comprise:

applying reverse register transformation with respect to the runtime event; and

determining if the instruction instance is valid from the applied reverse register transformation.

50. The computer program product of claim 47 wherein the memory reference object includes a physical memory reference object or a logical memory reference object.

51. The computer program product of claim 50 wherein the physical memory reference object includes cache, a cache line, a cache sub-block, a cache level, a memory controller, addressable memory, virtually addressable memory, physically addressable memory, or a memory-management page translation unit.

52. The computer program product of claim 50 wherein the logical memory reference object includes a source-level data object, a memory segment, a heap variable, a variable instance, and a stack variable.

53. The computer program product of claim 52 wherein the source-level data object includes a data type, a data type definition, an operand, a statically linked object, a data structure, or an expression.

54. The computer program product of claim 53 wherein the statically linked object includes a global variable or a static variable.

55. An apparatus comprising:

a processor;

memory; and

means for identifying a memory reference object and identifying a data address corresponding thereto from an instruction instance that corresponds to one or more runtime events, and aggregating a set of addresses that include the data address, based at least in part on the memory reference object.

56. The apparatus of claim 55 wherein the memory reference object includes a physical memory reference object or a logical memory reference object.

57. The apparatus of claim 56 wherein the physical memory reference object includes cache, a cache line, a cache sub-block, a memory controller, addressable memory, physically addressable memory, virtually addressable memory, or a memory-management page translation unit.

58. The apparatus of claim 56 wherein the processor includes event condition counters.